

---

# Algoritmos en Grafos

---

## Coloreo de Grafos, Stable Marriage

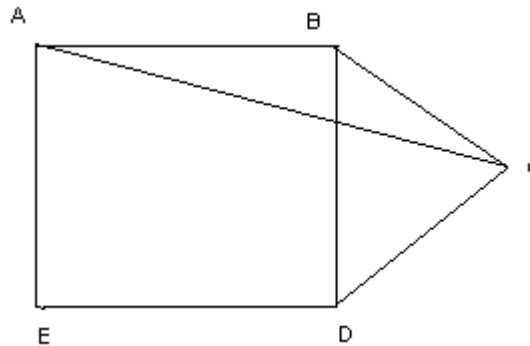
[Basado en el Material de Prof. Scott Hazelhurst, Ian Sanders y W.H. Tutte]

# Coloreo de Grafos

El coloreo de grafos es importante porque es una fuente importante de interés práctico y teórico

Observaremos el caso de coloreo de vértices

El coloreo de vértices significa asignar colores a los vértices



$C_1 = \{\text{rojo, azul, verde}\}$

$C_2 = \{1, 2, 3\}$

---

## Coloreo de Grafos

Un coloreo de un grafo es el coloreo de vértices así de que no hay dos vértices adyacentes que tengan el mismo color

Un coloreo que usa  $n$  colores es llamado *n-coloreo*

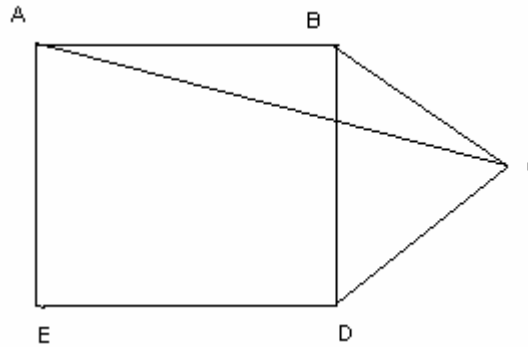
El mínimo número de coloreos e un coloreo de  $G$  es llamado el número cromático de  $G$  y se denota por  $\chi(G)$

Usualmente existen muchos modos de colorear un grafo

Definiciones auxiliares:

- Denotar el grafo completo de orden  $n$  como  $K_n$
  - Suponga que  $G=(V_1 \cup V_2, E)$  es un grafo bipartito  
 $G$  es un grafo completo bipartito si cada vértice en  $V_1$  es adyacente a cada vértice en  $V_2$ : si el número de vértices en  $V_1$  es  $r$  y el número de vértices en  $V_2$  es  $s$ , entonces se denota  $K_{r,s}$
-

# Coloreo de Grafos



1. Coloree el grafo
2. ¿Cuál es el número cromático de  $K_5$ ?
3. ¿Cuál es el número cromático de  $K_{3,3}$ ?

---

## Coloreo de Grafos

Decimos que un grafo  $G$  es  $k$ -coloreable si  $\chi(G) \leq k$

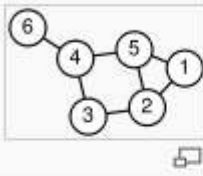

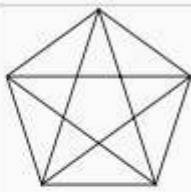

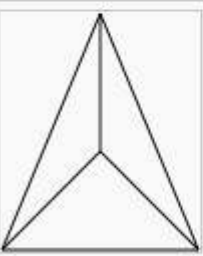

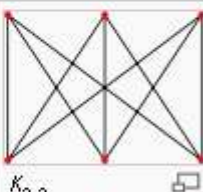

Teorema: Sea  $G=(V,E)$  un grafo entonces si el vértice con el mayor grado tiene grado  $r$ , entonces el grafo es  $(r+1)$ -coloreable

Prueba: Por Inducción en el número de vértices. Sea  $G$  un grafo

- Caso Base:  $n=0$ , el grado máximo es 0, así que el resultado es directo
  - Hipótesis Inductiva: Si el número de vértices en  $G$  es  $n$ , entonces el grafo es  $r+1$  coloreable
  - Paso Inductivo: Suponga que  $G$  tiene  $n+1$  vértices y sea  $v$  un vértice de grado  $r$
  - Borre  $v$  con los lados incidentes a este desde  $G$  para formar el grafo  $G'$
  - Existen  $n$  vértices que quedan en  $G'$ , por IH  $G'$  es  $r+1$  coloreable
  - Esto nos da cada vértice en  $G$  adyacente a  $v$  un color
  - Pero  $v$  tiene como máximo  $r$  vecinos de  $v$  en  $G$ , así que  $\exists$  un color para  $v$
-

# Coloreo de Grafos

Teorema: Cada grafo planar es 4-coloreable

Example graphs	
Planar	Nonplanar
 	 $K_5$ 
 The complete graph $K_4$ is planar 	 $K_{3,3}$ 

[tomado de Wikipedia]

---

## Coloreo de Grafos

Teorema: Cada grafo planar es 4-coloreable

Uno de los más famosos teoremas en matemáticas y ciencia de la computación

Fue planteado hace cien años – no existe una prueba concluyente

En los 70s, dos matemáticos mostraron que todos los grafos planares podrían ser representados por 1200 casos especiales

Y si cada uno de estos casos especiales podían ser 4-coloreados, entonces todos los grafos planares

Entonces, usando una computadora hicieron una búsqueda exhaustiva para encontrar el coloreo de cada uno de estos grafos especiales, por tanto demostrando que todos los grafos planares son 4-coloreables

Uno de los primeros ejemplos en los cuales se da la generación de una prueba por computadora con resultados desconocidos

Consecuencia del resultado - todos los mapas se pueden colorear con 4 colores

---

---

## Coloreo de Vértices y Complejidad

A pesar de que aparentemente es un problema simple, el coloreo de grafos es bastante difícil

El problema de los Tres Colores: Dado un grafo arbitrario  $G$ , determinar si es posible el colorearlo con tres colores

Teorema: El problema de los 3 colores es un problema NP-completo

---



---

## Aplicaciones del coloreo de grafos

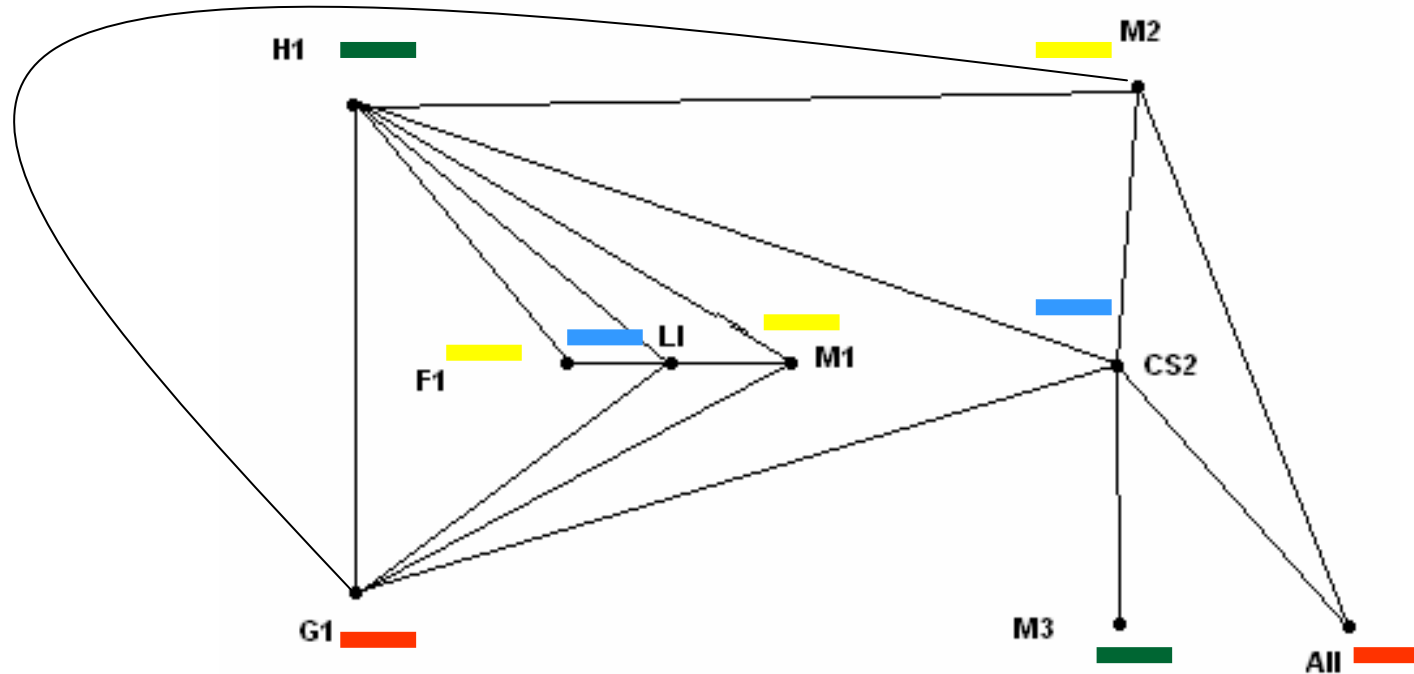
Calendarización: Seis estudiantes: A,B,C,D,E y F; cada uno tiene que dar un examen. Calendarizar los exámenes de tal modo que no haya cruce de horarios

- A da Cálculo II, Matemática II, Historia I
- B Cálculo II, Matemática II, Análisis II
- C Cálculo II y Matemática III
- D Historia I, Filosofía I, Literatura I
- E Geografía I, Historia I, Matemática I, Literatura I
- F Cálculo II, Matemática II y Geografía I

Solución – utilizar coloreo de grafos

- Crear un grafo con un vértice para cada examen
  - Existe un lado entre dos vértices si tienen un estudiante en común
-

# Aplicaciones del coloreo de grafos



---

## Aplicaciones del coloreo de grafos

Se logró encontrar un probable horario. Cada color corresponde a un lapso de tiempo en la tabla de exámenes:

Hora	0	1	2	3
	Historia I	Literatura I	Análisis II	Matemática I
	Cálculo II	Filosofía I	Geografía I	Filosofía I
				Matemática II

---

---

## Algoritmo de coloreo de grafos

No es posible el encontrar un algoritmo perfecto para el coloreo de grafos, pero existen diversos algoritmos para realizar un coloreo aproximado

```
AproxColor (V,E){  
    Ordenar los vértices de modo que  $\deg(v_i) \geq \deg(v_{i+1})$ ;  
    color( $v_0$ ) < 0  
    for (i=1; i < u; i++){  
        color( $v_i$ ) = Color índice más pequeño que no produce choques;  
    }  
}
```

---

---

## Algoritmo de coloreo de grafos

Se puede implementar el algoritmo planteado en un programa en Java.

Objetivo:

- Desarrollo de un método
- Ver las ideas en práctica

Dos secciones que se deben de implementar:

- Ordenamiento
  - Encontrar el menor color que no choque con los demás
-

---

## Ordenamiento

- Existen una gran cantidad de algoritmos para este propósito
- Se utilizará el ordenamiento por inserción: simple, no es tan eficiente

Idea básica:

- Mantener una lista a ser ordenada en dos partes: parte ordenada, parte no ordenada
  - Inicialmente, la parte ordenada contiene un elemento, la parte no ordenada contiene el resto de la lista
  - En cada etapa del algoritmo, se toma el primer elemento de la parte no ordenada y se inserta en la parte ordenada en la posición correcta
  - Al final, la parte ordenada contiene la lista completa, la parte no ordenada se queda vacía
-

---

# Ordenamiento

Especificación del código:

```
void insercion(int numeros[], int indice[])
```

- Inicialmente: *numeros* contiene un conjunto de enteros
  - Al final: *sindice* da permutaciones del índice en el array para que los números se ordenen. Es decir:
    - *sindice[0]* es el índice del elemento más grande en *numeros*
    - *sindice[n-1]* es el índice del elemento más pequeño en *numeros*
    - *for*  $0 \leq i \leq n-2$   
 $numeros[sindice[i]] \geq numeros[sindice[i+1]]$
-

---

## Ordenamiento

Bosquejo del código:

```
void insercion(int numeros[], int indice[]){  
    int siguiente;  
    indice[0]=0;  
    for (siguiente=1; siguiente < numeros.length; siguiente++){  
        insertar(numeros[siguiente]);  
    }  
}
```

- Invariante: La parte que denominamos *ordenada* siempre está ordenada.  
Al inicio sólo un elemento se encuentra en la parte ordenada
- Variante: La parte no ordenada se contrae después de cada iteración del algoritmo. El algoritmo se detiene cuando está vacío

Ambos garantizan la correctitud del algoritmo

---



# Ordenamiento

Implementación:

```
static void insercion(int nums[], int indice[]){
    int sig, actual, i;
    indice[0]=0;
    //Invariante
    //nums[indice[j-1]]>=nums[indice[j]]    0<=j<sig
    for (sig=1; sig < nums.length; sig++){
        actual=nums[sig];
        while (i>0 && nums[indice[i-1]] < actual){
            indice[i]=indice[i-1];
            i--;
        }
        indice[i]=sig;
    }
}
```

## Ordenamiento

Análisis del algoritmo: Suponga que hay  $n$  elementos en el arreglo

Bucle interno: Peor caso es cuando itera *siguiente* veces. El mejor caso es cuando nunca itera

De este modo, el costo en el peor caso es: *siguiente*  $\times c_1$ ; en el mejor caso es  $c_2$

El valor de *siguiente* está determinado por la iteración del bucle externo en la cual estemos - varía de 1 a  $n-1$

Si *siguiente*= $j$ , costo de iterar el bucle externo en el peor caso es  $jc_1 + c_3$  y en el mejor caso es  $c_2 + c_3$

■ Peor caso: Costo es:  $\sum_{j=1}^{n-1} jc_1 + c_3 = n(n-1)c_1/2 + nc_3 \approx n^2c_1/2 + nc_3 \in O(n^2)$

■ Mejor caso:  $\sum_{j=1}^{n-1} c_2 + c_3 = n(c_2 + c_3) \in O(n)$